CLAIMS

WHAT IS CLAIMED:

1. A method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop

determining dependencies between vector memory references in the loop, including determining unidirectional and circular dependencies; and

distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references.

2. A method, as set forth in claim 1, further comprising allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas.

3. A method, as set forth in claim 1, further comprising at least one section loop including the plurality of detail loops.

4. A method, as set forth in claim 1, wherein distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management.

5. A method, as set forth in claim 1, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

6. A method, as set forth in claim 1, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

7. A method, as set forth in claim 1, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

8. A method, as set forth in claim 1, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

9. A method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop

determining dependencies between vector memory references in the loop; and distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have dependencies therebetween are included in a common detail loop.

10. A method, as set forth in claim 9, further comprising allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas.

11. A method, as set forth in claim 9, further comprising at least one section loop including the plurality of detail loops.

12. A method, as set forth in claim 9, wherein distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management.

13. A method, as set forth in claim 9, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

14. A method, as set forth in claim 9, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

15. A method, as set forth in claim 9, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

16. A method, as set forth in claim 9, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

17. A method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop

determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops in response to cache behavior and the dependencies between the vector memory references in the loop.

18. A method, as set forth in claim 17, wherein distributing the vector memory references further comprises distributing the vector memory references into the plurality of detail loops with each loop having at least one of the identified vector memory references.

19. A method, as set forth in claim 17, further comprising determining dependencies between vector memory references in the loop, and wherein distributing the loop includes distributing the vector memory references into the plurality of detail loops,

wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop.

20. A method, as set forth in claim 17, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

21. A method, as set forth in claim 17, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

22. A method, as set forth in claim 17, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

23. A method, as set forth in claim 17, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

24. A computer programmed to perform a method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop

determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop.

25. A program storage medium encoded with instructions that, when executed by a computer, perform a method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop

determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop.

26. A compiler, comprising:

means for identifying a loop in a program;

means for identifying each vector memory reference in the loop

means for determining dependencies between vector memory references in the loop, including determining unidirectional and circular dependencies; and

means for distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references .

27. A compiler, as set forth in claim 26, further comprising means for allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas.

28. A compiler, as set forth in claim 26, wherein the means for distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management.

29. A compiler, as set forth in claim 26, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

30. A compiler, as set forth in claim 26, further comprising means for inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

31. A compiler, as set forth in claim 26, further comprising means for performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

32. A compiler, as set forth in claim 26, further comprising means for inserting at least one of a prefetch instruction and a cache management instruction into at least one of

said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

33. A method for reducing the likelihood of cache thrashing by software to be executed on a computer system having a cache, comprising:
executing the software on the computer system;
generating a profile indicating the manner in which the software uses the cache;
identifying a portion of the software using the profile data that may experience cache thrashing; and
modifying the identified portion of the software to reduce the likelihood of cache thrashing.

34. A method, as set forth in claim 33, wherein modifying the identified portion of the software to reduce the likelihood of cache thrashing further comprises:
identifying a loop in the identified portion of the software;
identifying each vector memory reference in the identified loop;
determining dependencies between the vector memory references in the identified loop of the software, including determining unidirectional and circular dependencies; and
distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references.

35. A method for reducing the likelihood of cache thrashing by software to be executed on a computer system having a cache, comprising:

executing the software on the computer system;

generating a profile indicating the manner in which the memory references of the software use the cache;

identifying a first and second portion of the memory references based on the profile, wherein the first portion of the memory references may be experiencing cache thrashing; and

distributing at least a portion of the first portion of the memory references into distinct loops, and placing at least the second portion of the memory references into the distinct loops.